XLBlocks

a Block-based Formula Editor for Spreadsheet Formulas

Jansen, Bas; Hermans, Félienne

DOI
10.1109/VLHCC.2019.8818748

Publication date
2019

Document Version
Accepted author manuscript

Published in
2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)

XLBlocks: a Block-based Formula Editor for Spreadsheet Formulas

Bas Jansen
Delft University of Technology
The Netherlands
Email: b.jansen@tudelft.nl

Felienne Hermans
Leiden University
The Netherlands
Email: f.f.j.hermans@liacs.leidenuniv.nl

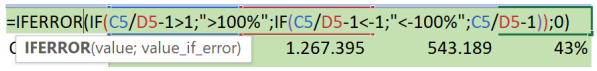
***Abstract*—Spreadsheets are frequently used in industry to support critical business decisions. Unfortunately, they also suffer from error-proneness, which sometimes results in costly consequences. Experiments in the field of program education have shown that programmers tend to make fewer errors and can better focus on the logic of a program if they use a block-based language instead of a textual one. We hypothesize that a block-based formula editor could support spreadsheet users in a similar way. Therefore, we develop XLBlocks and conduct a think-aloud study with 13 experienced spreadsheet users from industry. Participants are asked to create and edit several formulas, using our block-based language. We then ask them to evaluate this editor using the Cognitive Dimensions of Notations framework. We found that for all dimensions the block-based formula editor received a better evaluation than the default text-based formula editor.**

I. Introduction

Spreadsheets are widely used in industry. It has been estimated that 90% of all analysts in industry use spreadsheets for their calculations [1]. This observation indicates that spreadsheets are often used to support critical decisions. Unfortunately, almost all spreadsheets contain non-trivial errors [2]. As a consequence, decisions are based on incorrect information, which eventually can lead to significant loss of money[1]. To address these quality problems in spreadsheets, Hermans *et al.* researched the concept of code-smells in spreadsheet formulas and worksheets [3], [4]. That work was later extended by Cunha *et al.* [5]. Barowy *et al.* extended the concept of smells to also cover data [6].

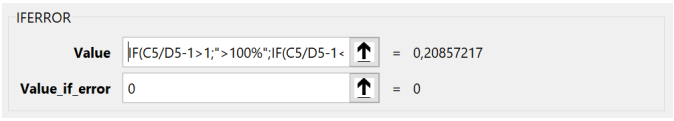
Concerning errors, Panko and Halverson distinguish three types of spreadsheet errors: mechanical (typing errors), logic (formula errors), and omission errors [7]. Panko also found that error rates for logic errors are higher than for mechanical errors, meaning, that most spreadsheet errors have their origin in formulas [8]. Therefore, although a spreadsheet model consists of data, layout, and formulas, we focus in this paper on creating and maintaining formulas in spreadsheets.

Unfortunately, the way the user interface facilitates the entering of formulas is rather limited. For example, in Microsoft Excel, formulas can be entered directly in a cell (see Fig. 1a), in the formula bar (see Fig. 1b), or created using the function wizard (see Fig. 1c).



(a) In-cell editing

(b) Editing in the formula bar



(c) Using the function wizard

Fig. 1. three ways for entering a formula in Excel

In all 3 of the editing methods, it is difficult to get an overview of the complete formula. For example, the double nested *if* formula in Figure 1b is represented as a string of characters on a single line. It is difficult to distinguish the two *ifs* from their arguments. The function wizard attempts to visualize this better but has the drawback that it only shows one function at a time. In Figure 1c the *iferror* functions is highlighted, but the *ifs* are 'hidden' in the value field of the *iferror* function. Especially with in-cell editing or using the formula bar, it is easy to forget or misplace a comma, parenthesis, or quote. In these cases, the user needs to know the exact syntax of the function, meaning, the order and purpose of the function's arguments and whether they are mandatory or optional.

These problems with formula syntax are similar to the challenges novice programmers encounter when they learn to program. Research has shown that block-based program languages improve the performance of novice programmers by minimizing the possibility of syntax errors and removing the necessity for accurate punctuation [9], [10].

We hypothesize that a block-based formula editor for spreadsheets could support spreadsheet users in a similar way. This paper introduces XLBlocks, a block-based formula editor for Excel and presents the results of a think-aloud study in which the participants perform a set of typical spreadsheet task

[1]http://www.eusprig.org/horror-stories.htm

978-1-7281-0810-0/19/$31.00 ©2019 IEEE

with XLBlocks. After the tasks, we interview them and ask them to evaluate XLBlocks using the Cognitive Dimensions of Notation (CDN) framework [11].

II. XLBlocks: A Block-Based Formula Editor

This paper examines to what degree a block-based formula editor could support professional spreadsheet users while developing or maintaining formulas. In this section, we provide an overview of XLBlocks (Figure 2). XLBlocks aims to 1) implement a block-based interface (Section II-A), 2) facilitate users to replicate formulas across rows or columns (Section II-B), and 3) introduce new functions that are easier to use than some of the built-in Excel functions (Section II-C).

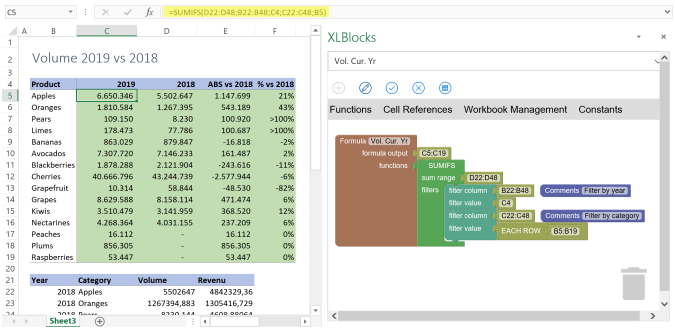


Fig. 2. Left the traditional screen with the textual formula highlighted at the top, right the block representation of the same formula

A. XLBlocks Interface

XLBlocks is an Excel Add-in developed with the Excel JavaScript API [12]. The Blockly library [13] was extended with custom blocks and a code generator for the definition and generation of spreadsheet formulas. Based on frequently used functions in the Enron corpus [14], we included the following functions in the research prototype: SUM, SUMIFS, IFERROR, INDEX, MATCH, VLOOKUP, IF, -, /, >, <. Videos demonstrating the user interface of XLBlocks are available online[2].

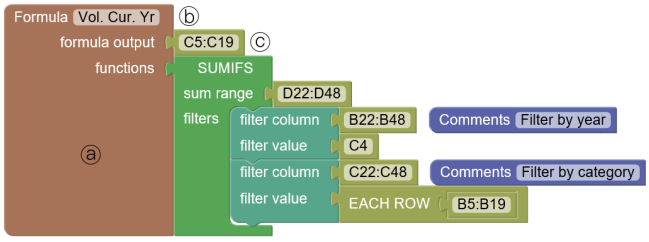


Fig. 3. Example of a block definition of a SUMIFS formula

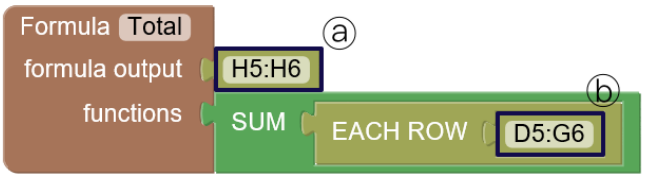
To use XLBlocks, a user starts the definition with a formula block (see (a) in Figure 3). The user can give the formula a name (b), has to specify the output range (c) (the cells in the spreadsheet that will receive this formula) and the functions that are used in the formula. In Figure 3 the SUMIFS function is used as an example. This function will calculate a conditional sum based on one or more criteria. Therefore, in XLBlocks, it is possible to connect one or more filter blocks as arguments to the SUMIFS function. In this example, two comments have been added to the formula to document the filters that are used in this sum. These comments are not transferred to the spreadsheet but will be available in the XLBlocks editor.

B. EACH ROW and EACH COLUMN

Within the spreadsheet paradigm, it is common practice to define a formula and then replicate it across many rows or columns by dragging it down or to the right. To facilitate this in the XLBlocks editor, we introduced two special blocks: 'EACH ROW' and 'EACH COLUMN'.

	AccountNr	Description	Q1	Q2	Q3	Q3	Total
5	8000	Sales Revenu	€ 3.922	€ 9.999	€ 1.135	€ 4.369	=SUM(D5:G5)
6	8100	Cost of Sales	€ -1.569	€ -4.600	€ -499	€ -2.534	=SUM(D6:G6)
8	Gross Profit		€ 2.353	€ 5.399	€ 636	€ 1.835	€ 10.223

(a) Row totals in excel



(b) EACH ROW block in XLBlocks

Fig. 4. Using EACH ROW to replicate formulas across multiple rows

Figure 4 shows an example of calculating row totals in cells H5 and H6. In Excel, a user would define the formula in cell H5 and copy it to H6. In XLBlocks it is sufficient to specify both cells H5 and H6 as output range (a) and use the 'EACH ROW' block to specify that the sum of each row in the range D5:G6 should be calculated (b). It also means that if the average instead of the sum should be calculated, this can be solved by editing only the formula definition in XLBlocks instead of editing a formula and dragging it down. The 'EACH COLUMN' block offers the same functionality for calculations across multiple columns.

C. Lookups

Excel has five functions that can be used to lookup data (similar to join functions in databases): VLOOKUP, HLOOKUP, INDEX, MATCH, and LOOKUP. They differ in syntax and arguments, and some have problematic defaults [15].

From these formulas, VLOOKUP is used the most frequently, although the combination of INDEX and MATCH is a better alternative. That combination can be used both horizontally and vertically, there are no sorting requirements, and the lookup range can be located anywhere. Unfortunately,

[2]https://doi.org/10.6084/m9.figshare.8863532

it is not frequently used because of it's complexity. It is an excellent candidate to explore if we can replace this with a new function that is easier to use. Therefore we define a general purpose 'LOOKUP' block in XLBlocks (Figure 5).

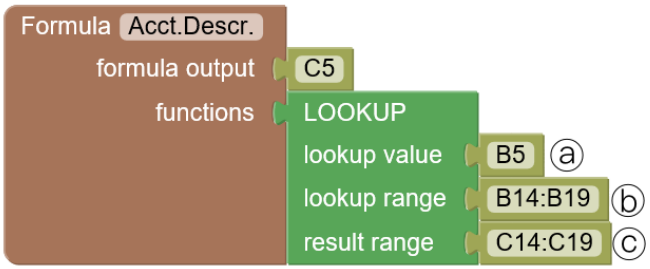


Fig. 5. LOOKUP formula in XLBlocks

For this function, the user simply specifies the value they are looking for (a), the range that contains the possible lookup values (b) and the range that contains the matching results (c). XLBlocks translates this into the following formula:

```
=INDEX(C14:C19,MATCH(B5,B14:B19,0))
```

Using the LOOKUP block in XLBlocks give users the flexibility of the INDEX and MATCH combination without the complexity.

III. Setup of Think-aloud Study

A. Research Questions

The goal of this paper is to examine to what degree a block-based formula editor could support professional spreadsheet users while developing or maintaining formulas. To address this goal, we develop XLBlocks, a block-based formula editor for Excel, and conduct a think-aloud study in which participants perform eight typical spreadsheet tasks. After the tasks, we interview the participants and ask them to evaluate the XLBlocks interface, using the CDN framework. For each dimension, we ask them to answer the two following research questions:

- *RQ1:* What are the benefits of XLBlocks regarding this dimension?
- *RQ2:* What are the drawbacks of XLBlocks regarding this dimension?

B. Participants

We invited thirteen professional Excel users by mail from nine different companies within our network of industrial partners (Table I). Included in the study were participants who use Excel professionally and who use formulas. All participants, except one, use Excel multiple times a day and have on average 20 years of experience using Excel. We asked them to asses their level of expertise with Excel on a ten point scale (one = low, ten = high). We used this scale because it is widely used in (European) schools to grade work and the participants are more familiar with it than for example a Likert scale. On average they rated themselves an eight out of ten.

TABLE I
Overview of think-aloud study's participants

Nr.	Gender	Age	Functional Domain	Excel level	Frequency	Experience (yrs)
P1	M	38	SE†	9	Daily	15
P2	M	54	SE	7	Daily	20
P3	M	51	SE	8	Daily	10
P4	M	62	Finance	8	Daily	25
P5	M	54	Operations	7	Daily	23
P6	M	25	Operations	7	Daily	4
P7	F	47	Finance	8	Daily	25
P8	M	53	Finance	9	Daily	25
P9	M	39	Finance	7	Daily	20
P10	M	50	CTO	4	Monthly	25
P11	M	56	Finance	8	Daily	25
P12	M	41	SE	8	Daily	20
P13	M	47	Finance	9	Daily	20
Average		47		8		20

† SE = Software Engineering

C. Think-Aloud Study

In a think-aloud study, we ask the participants to perform eight typical spreadsheet tasks in two different spreadsheets. The tasks are summarized in Table II.

TABLE II
Tasks to be performed by participants in think-aloud study

Task	Description
T1	Create row totals with SUM function
T2	Create column totals with SUM function
T3	Move output of column totals to different range
T4	Lookup account descriptions with LOOKUP function
T5	Create a conditional sum on two conditions with the SUMIFS function
T6	Change SUMIFS formula to calculate sum of a different range
T7	Move output of SUMIFS formula
T8	Explain a formula with a nested IF structure

To ensure that the tasks would be similar to the tasks the participants perform in their spreadsheets, we selected formulas that are frequently used in spreadsheets from the Enron corpus [14] and our collection of industry spreadsheets [16].

Spreadsheets are on average used by 13 different users [17], which implies that there are many moments a spreadsheet is transferred from one user to another. Therefore, we are interested to see if a block representation of a formula can also support a user in explaining the formula to somebody else, for which, we added task T8.

Before the participants start with the tasks, they receive a ten to fifteen-minute instruction about the XLBlocks interface. They get about 40 minutes to finish the tasks, and we ask them to think-aloud. We provided the participants with two sample spreadsheets and explained each task. The sample spreadsheets are available online[3]. If a participant created a block-model leading to an invalid formula, the formula would not be generated. When such errors occurred during the tasks, we explained the cause of the error.

[3]https://doi.org/10.6084/m9.figshare.8863532

Five participants performed the tasks individually, the remaining eight participants in pairs. All participants performed tasks T1 to T7, but we asked the participants that worked in pairs additionally to perform T8.

Because of the exploratory nature of the study, we did not measure the time participants needed to complete the tasks. Also, the participants were not quantitatively assessed in performing the tasks. The tasks enabled participants to get experience with XLBlocks so that they could reflect on usability and could provide us with feedback in the interviews.

D. Interview

After the tasks, we conduct a semistructured interview of about 30 to 40 minutes with each participant. We use the CDN framework to structure the interviews. Participants that perform the tasks in pairs are also interviewed in pairs. We transcribed all interviews, grouped the answers per dimension, complemented it with observations from the think-aloud study, and used the combined information to answer the two research questions per dimension.

The CDN framework has been used in several usability studies [18]–[21] and Blackwell and Green developed a questionnaire for the CDN [22].

For the interview, we used the dimensions as defined in [23]. We made two adjustments. We excluded the dimension *Abstraction Gradient* because users can not create their own blocks in XLBlocks. Furthermore, we added the dimension *Provisionality* as is described in [11], [24]. The dimension refers to the opportunities that users have to try different design options. Spreadsheets are continuously evaluated, and users are used to how easy it is to try something out. For that reason, we included this dimension. It is closely related to the dimension *Premature commitment*, and we discuss the two dimensions together.

We do not ask participants to fill out the CDN questionnaire, but rather, we use the questions to structure the interviews. It allows us to clarify a dimension if we notice that participants have difficulties understanding it. In addition, we can probe participants for additional details. Finally, for each dimension, we ask participants to grade the usability of both XLBlocks and the built-in Excel formula editor on a scale from 1 to 10.

IV. Results

This section describes the results of the interviews. An overview is given in Table III and Figure 6. XLBlocks received a better evaluation on all dimensions. The biggest difference is found on the dimension *Secondary notation* and according to the participants, the two interfaces score similarly on the dimension *Hard mental operations*. In the remainder of this section, we present the results of the user-study and interviews per dimension.

A. Diffuseness

The dimension *Diffuseness* describes the verbosity of the language. How many symbols and space is required to express meaning.

TABLE III
CDN Evaluation of XLBlocks and Excel's formula editor

Dimension	XLBlocks	Excel formula editor
Diffuseness	7.4	5.6
Role-expressiveness	7.9	6.3
Secondary notation	8.1	4.3
Viscosity	8.1	6.2
Visibility	8.2	5.6
Closeness of mapping	7.3	5.6
Consistency	7.6	5.4
Error-proneness	7.6	5.1
Hard mental operations	7.4	6.4
Hidden dependencies	7.8	5.6
Premature commitment & Provisionality	7.9	5.0
Progressive evaluation	8.2	4.8

Average score per dimension on a ten point scale.

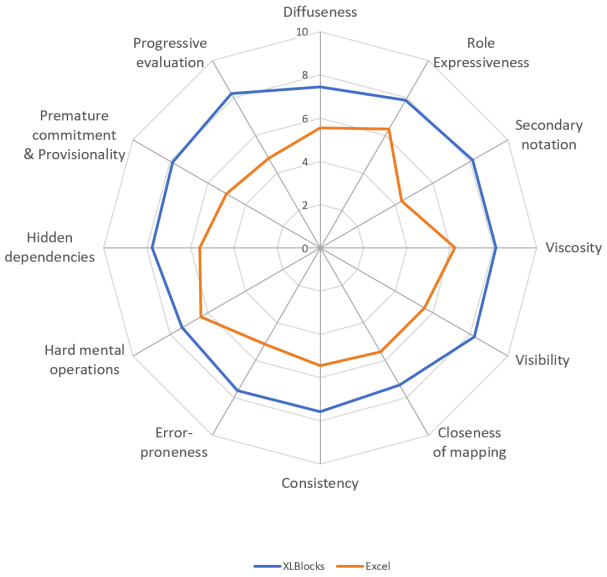


Fig. 6. Evaluation of formula editor in XLBlocks and Excel on a ten point scale

RQ1: Benefits XLBlocks: When block-based languages (BBL) are compared to text-based languages, diffuseness is often seen as a drawback for the BBL. They tend to be much more verbose and need more physical space to express the same. However, when we asked our participants about the diffuseness, they considered it as a benefit that more space was available. Excel does not feature a fully equipped integrated development environment (IDE) for editing formulas, but provide the user with a formula bar that is very limited in space (see Figure 1b). The additional space the XLBlocs interface provides, makes it easier to read formulas, and because of the graphical nature of the BBL, it is easier to see and recognize the structure of the formula.

RQ2: Drawbacks XLBlocks: Some participants (P2, P8, P11, P12) indicate that the XLBlocks hides parts of the spreadsheet

interface, which could be a problem in the case of larger spreadsheets. However, they still prefer the additional space to inspect and understand the formula. Participants P1 and P2 noticed that in the XLBlocks interface when additional functions or arguments are added, formulas tend to grow in width. They would prefer if formulas grow mainly in height.

B. Role-expressiveness

Role-expressiveness concerns the ease of seeing how a part of a formula relates to the formula as a whole. What is the meaning of every part of the formula?

RQ1: Benefits XLBlocks: According to the participants, the meaning of every individual block is clear. As a reason, they name the text labels on every block. For function names that is obvious, but also the arguments of every function are named (a, b, and c in Figure 5). This is not the case in the formula bar, and according to the participants, it makes the reading of the formula much easier. Also the consistent use of colors (all functions have the same color, see also Section IV-F: Consistency) support this.

RQ2: Drawbacks XLBlocks: On the other hand, when we observed the participants executing the requested tasks, it became clear that the meaning of the special blocks 'EACH ROW' and 'EACH COLUMN' was not as intuitive as we had hoped. Also the term 'output range' lead to some confusion. Especially, in combination with the 'LOOKUP' function that contained an argument that was named 'Result range' (See also c in Figure 5). Also, although constants have a different color than functions, participants P9 and P10 interpreted the constant block that was used to indicate a constant number as the function number (which does not exist in Excel).

C. Secondary Notation

Secondary notation expresses the options user have to convey additional information about a formula that is not part of the formal syntax of the formula. Spreadsheets as a whole provide ample opportunity for secondary notation. Therefore, it is important to note that we are comparing XLBlocks not with the spreadsheet, but with the formula editor of Excel.

RQ1: Benefits XLBlocks: All participants indicated that it is straightforward to add secondary notation to the formula definition in XLBlocks. XLBlocks has special comment blocks that can be dragged anywhere in the formula. In the excel formula bar, it is not possible to add comments. However, a comment can be assigned to the cell that contains the formula. According to the participants that is not optimal because 1) one can only assign one overall comment for the complete formula, 2) the comment is visible in the spreadsheet model and 3) if a formula has been copied across multiple rows or columns, it is not clear to which formulas the comment belongs.

RQ2: Drawbacks XLBlocks: For secondary notation, the participants, did not mention any specific drawbacks. Participant P2 remarked that further improvement might be possible if a text field was added to the brown formula block (See Figure 5) that could be used to write some general information about the formula. On the other hand participants, P1 and P2 asked themselves if they would ever use the comments: *"Because the block definition of the formula is easier to read, the formula documents itself."*

D. Viscosity

The dimension *viscosity* measures the effort that is required to perform a single change.

RQ1: Benefits XLBlocks: The participants agree that changing a formula is simple in XLBlocks. This was confirmed by the fact that none of the participants made an error in the tasks that involved changing existing formulas. In comparison with the formula bar, it is easy to locate the part of the formula that one wants to change. Also, in the formula bar, users have to be very careful with the use of parentheses, commas, and quotes. This is something that is taken care of automatically in XLBlocks. It was also stated that with XLBlocks it is possible to change a group of formulas with a single edit.

RQ2: Drawbacks XLBlocks: Two participants (P12 and P13) comment that, when editing, they use search and replace a lot. This is possible in the formula bar but not in XLBlocks. Also, when a formula is not very complex, and an edit comprises only changing a few characters, it is much quicker to edit directly in the formula bar than in XLBlocks.

E. Closeness of Mapping

The dimension *Closeness of mapping* expresses the mapping between the programming world and the problem world.

RQ1: Benefits XLBlocks: All participants agree that XLBlocks visualizes the calculation that will be generated by Excel. Especially, when the formula is complex or nested this is visualized better in XLBlocks than in the formula bar (see Figure 7). Also, XLBlocks needs less interpretation. For example, the IF function has three named arguments in XLBlocks: *if*, *then*, and *else*. In Excel, the arguments are not named. It is by convention that the user knows that the first argument is the condition, the second the *then* statement, and the third the *else* statement. Also, the user has to know that commas separate these arguments. In XLblocks, the user does not need to know these conventions.

RQ2: Drawbacks XLBlocks: One participant (P4) argued that the XLBlocks interface has a less direct mapping to the problem the user tries to solve. The EACH ROW and EACH COLUMN blocks add another layer of abstraction, while in the formula bar the user is only editing one cell. The fact that a group of formulas is edited instead of a single formula reduces the closeness of mapping.

F. Consistency

Consistency means that parts of a formula that have a similar meaning should have a similar appearance.

RQ1: Benefits XLBlocks: Most participants perceived the block notation in XLBlocks as consistent. They name the use of color as the main reason. In XLBlocks each category of blocks has its own color. It is easy to see what is a function, cell reference, or constant. In the formula bar, this is less clear. Although Excel also colors the cell references, there

is no difference in typographic style between functions and constants (See Figure 1b).

RQ2: Drawbacks XLBlocks: In its current implementation, all functions in XLBlocks have the same color (green). This could result in one big green block of functions if a formula consists of several nested functions. Therefore, one participant (P2) suggested to color the functions according to their category (e.g., text functions would have a different color than math functions).

G. Error-proneness

The dimension *Error-proneness* indicates how easy it is to make errors while writing a formula.

RQ1: Benefits XLBlocks: In XLBlocks, the code generator inserts commas, parentheses, and quotes at the right place in the formula. According to several participants (P4, P7, P9, P10, and P12), this is the main reason that fewer errors are made in XLBocks. Also, two participants (P3 and P13) observed that the function blocks in XLBlocks guide the user through the function. As a consequence, it is not possible that arguments are forgotten or used in the wrong order.

RQ2: Drawbacks XLBlocks: During the tasks, we observed participants placing blocks in the wrong position. This corresponds with feedback we received from other participants (P9, P10, and P12) that it was possible to misplace blocks. Also, a participant forgot to give the formula a name. Although this does not lead to an error in the formula, it makes it more challenging to find the formula.

H. Hard Mental Operations

Working on a formula requires mental effort. The dimension *Hard mental operations* indicates to which extent the block-based language used in XLBlocks itself causes this mental effort.

RQ1: Benefits XLBlocks: The participants stated that in XLBlocks they had to think less about the exact syntax of the formula, the correct order of the arguments and when nesting functions, the order of the functions themselves.

RQ2: Drawbacks XLBlocks: For the participants, the concept of the 'EACH ROW' and 'EACH COLUMN' blocks was the most difficult to understand. Partially this is not something caused by the notation, but by the concept itself. On the other hand, participants had difficulties placing the 'EACH ROW' block at the right spot. They could use more guidance from the XLBlocks interface.

I. Hidden Dependencies

Spreadsheets are disreputable for their hidden dependencies. However, they are less common in a spreadsheet formula. A hidden dependency in a spreadsheet could be a function and its corresponding arguments. Consider, for example, the following formula:

```
=IFERROR(IF(C5/D5-1>1,">100%",IF(
 C5/D5-1<-1,"<-100%",C5/D5-1)),0)
```

The zero at the end of the formula is an argument of the IFERROR function at the beginning of the formula and could be considered as a hidden dependency.

RQ1: Benefits XLBlocks: According to the participants, it is much easier to see hidden dependencies in a formula in XLBlocks than in the formula bar in Excel. They name the previously mentioned IFERROR function as an example (Figure 7). Because the 'in case of error' argument is physically connected to the IFERROR function, the dependency is much more visible.

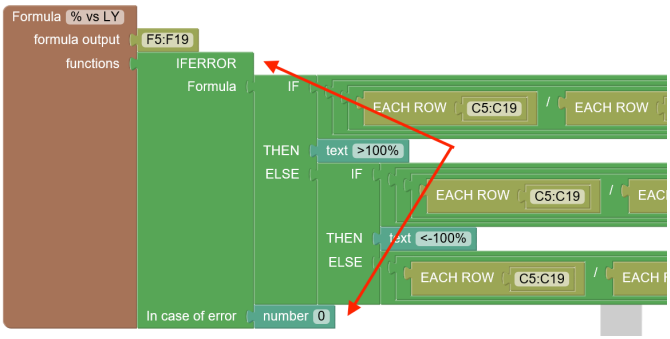


Fig. 7. Example of the IFERROR function, the 'in case of error' argument is visually connected to the function

RQ2: Drawbacks XLBlocks: The participants did not mention any drawbacks of XLBlocks concerning *Hidden dependencies*.

J. Premature Commitment & Provisionality

The dimension *Premature commitment* expresses the extent to which the user is forced to take decisions before the necessary information is available. This dimension was first defined in [23]. Later it was extended in [24] and [11] with a separate dimension *Provisionality* that concerns the opportunities for the user to play around with ideas. Is it, for example, easy to try different design options or to use 'what-if' scenarios?

RQ1: Benefits XLBlocks: According to the participants, the user has complete freedom in the order that blocks are dragged onto the canvas. During the tasks, we observed that dragging out the necessary blocks helped the participants to think about the formula. They said things like *"I will start with the blocks I know ..."* (P8) and *"Now I start seeing how it will come together ..."* (P7). They compared it with the formula bar where users are forced to start with the outer function and work their way inwards. Participants also recognized that it was possible to create two variants of a part of the formula and that they could easily swap them to try out the different options. They liked that it was not necessary to 'clean up your canvas'.

RQ2: Drawbacks XLBlocks: The participants did not mention any drawbacks of XLBlocks concerning *Premature commitment* and *Provisionality*.

K. Progressive Evaluation

While developing a formula, users often want to check if the formula gives the desired results. The dimension *Progressive evaluation* focuses on how a notation facilitates this.

RQ1: Benefits XLBlocks: The participants perceived the way XLBlocks makes it possible to check the formula during

development, more comfortable to work with than the Excel formula bar. As main reason, they named the blocking error they get in Excel when the formula is not correct.

RQ2: Drawbacks XLBlocks: One participant would prefer if XLBlocks showed a real-time version of the formula that is being generated.

V. Discussion

A. Learnability

We know that block-based languages perform well on learnability [25], [26]. This was confirmed in our think-aloud study. After receiving only 15 minutes of instruction, all participants were able to finish the tasks. Several participants (P1, P2, P7, and P8) remarked that it surprised them how easy it was to learn to work with XLBlocks.

We also observed that the participants were able to create a formula with the *lookup* function in XLBlocks, which does not exist in Excel. It shows that even if the function is unknown, the blocks are intuitive enough that users can work with it.

B. Further Reduce Error-Proneness

According to the participants, it is less likely to make errors in XLBlocks than in the formula bar of Excel. Nevertheless, based on the feedback we received during the think-aloud study, it can be further improved. In the current version of XLBlocks, it is allowed to connect invalid combinations of blocks (e.g., connecting a function block where a cell reference block is expected). This can be solved if more robust type-checking is implemented in XLBlocks.

Users could be further supported by a preconfigured group of blocks. Instead of dragging a formula block and a range block from the toolbox and connect them on the canvas, they can drag a preconfigured formula block with the output range already attached. Blockly offers this feature with shadow blocks (placeholder blocks) or block groups (Figure 8) [27].

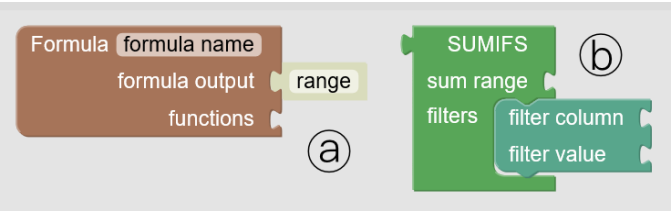


Fig. 8. An example of a formula block with a *shadow* range block (a) and a block group consisting of A SUMIFS block and a filter block (b).

Inspired by this concept, some participants suggested the idea to create your own templates of blocks for formula patterns that you use frequently and store them on your toolbox (similar to the backpack feature in Scratch [28]).

C. Simultaneous Use of XLBlocks and the Formula Bar

XLBlocks has been developed as an alternative way to edit a formula in Excel. It is not meant to replace the formula editor of Excel but to complement it. XLBlocks present the block-based formula embedded into the spreadsheet itself. Some participants (P10 and P13) named this as one of the strengths of XLBlocks. According to them some tasks, like making small edits in simple formulas, are better suited for the formula bar, while other tasks like editing complex formulas or explaining a formula to somebody else are better suited for XLBlocks.

With XLBlocks it is possible to develop a block-based specification of a formula and generate a valid spreadsheet formula. The other direction: generating a block-based formula from a spreadsheet formula, is not yet implemented. Several participants would like to see this feature added. Especially when explaining complex formulas to others, it would be convenient if the user could click on the formula and see the block-based representation in XLBlocks and use that as a basis for the explanation.

D. Threats to Validity

A threat to the external validity of our evaluation concerns the representativeness of the participants. Future studies are necessary to generalize our findings.

Another threat to the external validity of our evaluation is the representativeness of the tasks the participants had to perform. To mitigate this, we used formulas that professional spreadsheet users use in real-life spreadsheets.

We did not randomly select our participants, which is a threat to the internal validity. Nevertheless, we believe the group of participants serve as a useful reference group since they all work with spreadsheets daily and have on average 20 years of experience using Excel. Furthermore, they work at different companies, have different backgrounds, and work in different functional domains.

We are both the designers of XLBlocks and the interviewers, and this is another internal threat to the validity of our evaluation. We minimized this threat to use the CDN framework to guide the questions in the interview and making sure that all aspects of the usability of the XLBlocks interface were covered.

VI. Related Work

A. Spreadsheets and Visual Languages

Most related to our research is the work of Burnet *et al.* [29], Leitão and Roast [30], Sarkar *et al.* [31], and Abraham *et al.* [21].

Burnett *et al.* developed Forms/3, a general purpose visual language that builds upon the spreadsheet paradigm. They leveraged the visual aspect of spreadsheets and at the same time tried to overcome some of the spreadsheet limitations like a limited number of types and the lack of abstraction capabilities. However, they consider the spreadsheet as a whole as a visual language, while in this paper, we focus on a visual language for the formula editor.

Leitão and Roast designed a visual language to represent spreadsheet formulas graphically. They developed two variants: an 'Explicit Visualization' (EV) and a 'Dataflow Visualization' (DV). In both variants, numeric values, cell references, strings, operators and built-in spreadsheet functions

are represented in different combinations of shape and color. In EV, the visualized formula is a visual match of the original textual formula, while in DV the formula is presented hierarchically in a syntax tree. Both are inspired by dataflow diagrams and are less textual than a block-based language.

Sarkar *et al.* propose multiple-representation editing in spreadsheets. They introduce Calculation View, an alternative representation of the spreadsheet, primarily designed for viewing formulas and their groupings. They use a new textual syntax for copying a formula into a block of cells and naming cells or ranges. Calculation View uses a textual notation in a columnar grid of pseudocells to maintain similarity with the spreadsheet grid.

Abraham *et al.* introduced ViTSL, a visual specification language for spreadsheets. The language allows the definition of spreadsheet templates that can be used by a spreadsheet generator to create Microsoft Excel spreadsheets automatically [32]. Derived from ViTSL, Engels and Erwig developed ClassSheets [33]. A ClassSheet represents both the structure and relationships of the involved (business) objects within the spreadsheet. It narrows the semantic distance between a problem domain and a spreadsheet application. A drawback of this approach was the lack of connection between the standalone model development environment where the ClassSheet was defined and the spreadsheet itself. As a result, automatic synchronization between the model and the spreadsheet was not possible. Cunha *et al.* [34] embedded the ClassSheet in the spreadsheet itself and made co-evolution of the model and the spreadsheet possible.

B. Cognitive Dimensions of Notation

Green and Petre have used the CDN framework as an evaluation technique for visual programming environments [23]. It provides a vocabulary for discussing the usability of programming languages. In their study, they present an outline of the cognitive dimensions and use them to evaluate two different visual programming environments.

The CDN were also used to design questionnaires intended for users to evaluate the usability of programming tools. In reaction, Blackwell and Green developed a generalized questionnaire and conducted a pilot study that used the questionnaire with a wide range of respondents [22]. The results of that study showed that the CDN questionnaire is a suitable tool for user evaluation of programming languages, tools, and environments.

The framework has been used to evaluate multiple programming languages. Most related to our research is the previously mentioned research of Abraham *et al.*. They used the CDN to evaluate their visual specification language ViTSL.

C. Block-Based Languages

Glinert introduced in 1986 the language BLOX, which can be considered as the first block-based language [35]. Research into block-based languages increased after the introduction of languages like Alice [36], Scratch [10], and Blockly [13]. These languages were designed as programming environments for younger learners.

Related to our research is the work of Weintrop *et al.* [37]. They created CoBlox, a block-based interface for programming a one-armed industrial robot. They showed that block-based programming could make a complex task like programming an industrial robot accessible for adults with limited programming experience.

Also related to our research is the study of Holwerda and Hermans [38]. They conducted a user study with Ardublockly, a block-based language derived from Blockly. In their study, they focus on gaining an understanding of the strengths and weaknesses of block-based languages as seen by professionals.

VII. Concluding Remarks

This paper aims to explore if editing spreadsheet formulas can be eased by using a block-based formula editor. We, therefore, developed XLBlocks, a block-based formula editor, and conducted a think-aloud study in which we evaluated the usability of the editor using the CDN framework. XLBlocks received on all dimensions a better evaluation. The difference was the most notable for the dimensions *Secondary notation*, *Error-proneness*, *Progressive evaluation*, *Premature commitment* and *Provisionality*.

Users recognized that in XLBlocks, they do not have to consider the correct syntax of functions. Also editing a part of a formula is easier in XLBlocks because they can easily drag and drop different parts of the formula on the canvas. During the think-aloud study, we observed that dragging different blocks on the canvas also supported the user in thinking about the formula. They started with the blocks they were sure about and then focused on the more difficult parts. They appreciated that they could freely decide in which order they would build the formula.

We also received feedback on areas that could be further improved. Varying the colors of the function block (e.g., by category) could make the formulas easier to read. In the current prototype, it is possible to connect invalid combinations of blocks. This could be solved by more robust type-checking and provide block groups as templates.

This paper gives rise to several directions for future work. At the moment, XLBlocks can generate a valid Excel formula from a block-based specification. We will extend the prototype with the possibility to generate a block-based representation of a spreadsheet formula. Furthermore, we are planning to make XLBlocks aware of the structural changes a user makes to the spreadsheet like inserting or deleting rows and columns. Finally, we will explore the possibility to facilitate the user to create templates for formulas that they use frequently.

References

[1] W. Winston, "Executive education opportunities millions of analysts need training in spreadsheet modeling, optimization, monte carlo simulation and data analysis," *OR MS TODAY*, vol. 28, no. 4, pp. 36–39, 2001.

[2] R. R. Panko, "Spreadsheet errors: What we know. what we think we can do," *arXiv preprint arXiv:0802.3457*, 2008.

[3] F. Hermans, M. Pinzger, and A. v. Deursen, "Detecting and visualizing inter-worksheet smells in spreadsheets," in *Proceedings of the 2012 International Conference on Software Engineering*. IEEE Press, 2012, pp. 441–451.

[4] F. Hermans, M. Pinzger, and A. Deursen, "Detecting code smells in spreadsheet formulas," *Proceedings of the International Conference on Software Maintenance (ICSM)*, 2012.

[5] J. Cunha, J. P. Fernandes, H. Ribeiro, and J. Saraiva, "Towards a catalog of spreadsheet smells," in *Computational Science and Its Applications–ICCSA 2012*. Springer, 2012, pp. 202–216.

[6] D. W. Barowy, D. Gochev, and E. D. Berger, "Checkcell: Data debugging for spreadsheets," in *ACM SIGPLAN Notices*, vol. 49, no. 10. ACM, 2014, pp. 507–523.

[7] R. R. Panko and R. P. Halverson, "Spreadsheets on trial: a survey of research on spreadsheet risks," in *Proceedings of HICSS-29: 29th Hawaii International Conference on System Sciences*, vol. 2, Jan 1996, pp. 326–335 vol.2.

[8] R. R. Panko, "What we know about spreadsheet errors," *Journal of Organizational and End User Computing (JOEUC)*, vol. 10, no. 2, pp. 15–21, 1998.

[9] T. W. Price and T. Barnes, "Comparing textual and block interfaces in a novice programming environment," in *Proceedings of the eleventh annual International Conference on International Computing Education Research*. ACM, 2015, pp. 91–99.

[10] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: Programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1592761.1592779

[11] A. F. Blackwell, C. Britton, A. Cox, T. R. Green, C. Gurr, G. Kadoda, M. Kutar, M. Loomes, C. L. Nehaniv, M. Petre *et al.*, "Cognitive dimensions of notations: Design tools for cognitive technology," in *International Conference on Cognitive Technology*. Springer, 2001, pp. 325–341.

[12] Javascript API for Office. Accessed: 2019-04-28. [Online]. Available: https://docs.microsoft.com/en-us/office/dev/add-ins/reference/javascript-api-for-office

[13] N. Fraser, "Ten things we've learned from blockly," in *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, Oct 2015, pp. 49–50.

[14] F. Hermans and E. Murphy-Hill, "Enron's spreadsheets and related emails: A dataset and analysis," in *Proceedings of the 37th International Conference on Software Engineering-Volume 2*. IEEE Press, 2015, pp. 7–16.

[15] F. Hermans, E. Aivaloglou, and B. Jansen, "Detecting problematic lookup functions in spreadsheets," in *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Oct 2015, pp. 153–157.

[16] E. Aivaloglou, D. Hoepelman, and F. Hermans, "Parsing excel formulas: A grammar and its application on 4 large datasets," *Journal of Software: Evolution and Process*, vol. 29, no. 12, p. e1895, 2017, e1895 smr.1895. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1895

[17] F. Hermans, "Analyzing and visualizing spreadsheets," Ph.D. dissertation, PhD thesis, Software Engineering Research Group, Delft University of Technology, Netherlands, 2012.

[18] M. Bellingham, S. Holland, and P. Mulholland, "A cognitive dimensions analysis of interaction design for algorithmic composition software," 2014.

[19] M. Kauhanen and R. Biddle, "Cognitive dimensions of a game scripting tool," in *Proceedings of the 2007 conference on Future Play*. ACM, 2007, pp. 97–104.

[20] F. Turbak, D. Wolber, and P. Medlock-Walton, "The design of naming features in app inventor 2," in *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2014, pp. 129–132.

[21] R. Abraham, M. Erwig, S. Kollmansberger, and E. Seifert, "Visual specifications of correct spreadsheets," in *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*. IEEE, 2005, pp. 189–196.

[22] A. F. Blackwell and T. R. Green, "A cognitive dimensions questionnaire optimised for users." in *PPIG*, vol. 13, 2000.

[23] T. R. G. Green and M. Petre, "Usability analysis of visual programming environments: a 'cognitive dimensions' framework," *Journal of Visual Languages & Computing*, vol. 7, no. 2, pp. 131–174, 1996.

[24] T. Green and A. Blackwell, "Cognitive dimensions of information artefacts: a tutorial," in *BCS HCI Conference*, vol. 98, 1998, pp. 1–75.

[25] D. Weintrop and U. Wilensky, "To block or not to block, that is the question: Students' perceptions of blocks-based programming," in *Proceedings of the 14th International Conference on Interaction Design and Children*, ser. IDC '15. New York, NY, USA: ACM, 2015, pp. 199–208. [Online]. Available: http://doi.acm.org/10.1145/2771839.2771860

[26] D. Bau, J. Gray, C. Kelleher, J. Sheldon, and F. A. Turbak, "Learnable programming: Blocks and beyond," *CoRR*, vol. abs/1705.09413, 2017. [Online]. Available: http://arxiv.org/abs/1705.09413

[27] Blockly Toolbox Guide. Accessed: 2019-05-06. [Online]. Available: https://developers.google.com/blockly/guides/configure/web/toolbox#shadow_blocks

[28] Scratch Backpack. Accessed: 2019-05-06. [Online]. Available: https://en.scratch-wiki.info/wiki/Backpack

[29] M. M. Burnett, J. W. Atwood, R. W. Djang, J. Reichwein, H. J. Gottfried, and S. Yang, "Forms/3: A first-order visual language to explore the boundaries of the spreadsheet paradigm," *Journal of functional programming*, vol. 11, no. 2, pp. 155–206, 2001.

[30] R. Leitão and C. Roast, "Developing visualisations for spreadsheet formulae: towards increasing the accessibility of science, technology, engineering and maths subjects," in *9th Workshop on Mathematical User Interfaces*, 2014.

[31] A. Sarkar, A. D. Gordon, S. P. Jones, and N. Toronto, "Calculation view: multiple-representation editing in spreadsheets," in *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Oct 2018, pp. 85–93.

[32] M. Erwig, R. Abraham, I. Cooperstein, and S. Kollmansberger, "Automatic generation and maintenance of correct spreadsheets," in *Proceedings of the 27th international conference on Software engineering*. ACM, 2005, pp. 136–145.

[33] G. Engels and M. Erwig, "Classsheets: automatic generation of spreadsheet applications from object-oriented specifications," in *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*. ACM, 2005, pp. 124–133.

[34] J. Cunha, J. Mendes, J. Saraiva, and J. P. Fernandes, "Embedding and evolution of spreadsheet models in spreadsheet systems," in *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Sep. 2011, pp. 179–186.

[35] E. P. Glinert, "Towards second generation interactive graphical programming environments," in *Proceedings of IEEE Workshop onVisual Language. IEEE CS Press, Silver Spring, MD*, 1986, pp. 61–70.

[36] M. Conway, R. Pausch, R. Gossweiler, and T. Burnette, "Alice: a rapid prototyping system for building virtual environments," in *CHI Conference Companion*. Citeseer, 1994, pp. 295–296.

[37] D. Weintrop, A. Afzal, J. Salac, P. Francis, B. Li, D. C. Shepherd, and D. Franklin, "Evaluating coblox: A comparative study of robotics programming environments for adult novices," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18. New York, NY, USA: ACM, 2018, pp. 366:1–366:12. [Online]. Available: http://doi.acm.org/10.1145/3173574.3173940

[38] R. Holwerda and F. Hermans, "A usability analysis of blocks-based programming editors using cognitive dimensions," in *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Oct 2018, pp. 217–225.